GAUSSIAN PBS TUTORIAL

For more details, email sv4e@virginia.edu

PART 1

Running Gaussian98 on the ITC Linux Clusters at UVA:
You will use the PBS (Portable Batch System) resource management software to run the code on the Linux cluster.

(1) Before you begin, make sure you have a research computing account; this is needed to access the Linux clusters. If you do not have an account, you will need to request for one here : http://www.itc.virginia.edu/research/hpc-account/getacct.html
(2) Use slogin or ssh to login to the cedar.itc.virginia.edu machine. When you logon to the cluster, you will be in your home directory. Create a directory for running the Gaussian98 programs, into which you will need to copy the Gaussian input file, benzene.gjf.
(3) Once you are in this directory, you need to create a new file called a batch job command file. This is a shell script containing the set of commands you want run on some set of cluster compute nodes. It also specifies the characteristics (attributes), and resource requirements (e.g. number of compute nodes and maximum runtime) that your job needs. A sample script file called gauss.sh is shown in Part 2 of this tutorial. Copy this to your Gaussian directory. You will need to use a UNIX text editor (e.g. vi) to edit the script files. Change the account name and email address to your own. You will need to make sure the name of the Gaussian input file is correct in the script file.
(4) Now submit the script file for execution on the cluster using the qsub command as follows:
lc2: /home/sv4e/Gaussian $ qsub gauss98.sh. When you submit the job, PBS returns a job id number as an example: 1234.lc2 .You will need this to check status, etc.
(5) To check status of your job, use the command 'qstat 1234' (or 'qstat –f 1234' for more details about the job). To check status of all jobs submitted to the cluster, use the command 'qstat –a'. To delete a job, us the command 'qdel 1234'.
(6) The files created are benzene.out, benzene.chk and the log file will be logfile_benzene.

PART 2

PBS JOB COMMAND (SCRIPT) FILE
Name it gauss98.sh , change account name, email address, and path to directory.

```
#!/bin/sh

#------------------- Start PBS declarations -------------------------#
# Name of job--can be anything
#PBS -N benzene

#Account to run job under
#PBS -A sv4e

#Send mail when the job terminates--change to your email address
#PBS -m ae
#PBS -M sv4e@virginia.edu

#Write output to "logfile"; append any error messages
#PBS -o logfile_benzene
#PBS -j oe

#PBS -l walltime=168:00:00
#PBS -l nodes=1:ppn=1
#-------------------- End PBS declarations --------------------------#


#------------------------ VARIABLES ---------------------------------#
cd $PBS_O_HOME
BASENAME="benzene"
echo "BASENAME is " $BASENAME
#BASENAME=*.gjf
#------------------------ End of VARIABLES --------------------------#



#------------------------ Start script ------------------------------#

ECHO=/bin/echo
CAT=/bin/cat
CP=/bin/cp
PERL=/usr/bin/perl
G98=/uva/bin/g98
MODULE="/opt/Modules/3.1.6/bin/modulecmd sh"
```

```
#############################################################
########
# Cleanup() #
# --------- #
# Put all exit stuff here! #
# #
# If called with one argument, then it exits with that argument. #
# If called with no arguments, then it exits zero. #
#############################################################
########
Cleanup()
{
   ##########################
   ##########################
   outdir=/home/sv4e/Trial_Gaussian
   # $CP -p $outdir/*.chk $PBS_O_WORKDIR/



   $ECHO ""
   $ECHO ""
   $ECHO
"############################################################
##"
   $ECHO "Job finished at `date`"
   $ECHO
"############################################################
##"
   $ECHO ""
   $ECHO ""


   if [ $# -eq 1 ] ; then
      # Note this means that it will exit with the first argument
      # If you call this with 0, then it will look like the script
      # was successfully run. This is NOT usually the case.
      # Calling Cleanup 0 may mess up your other jobs if they are
      # using the PBS wait commands.

      exit $1
   fi

   exit 0
}



#####
# The trap command catches the signal to kill the script.
# Instead of dying right away, we instead call the Cleanup
# subroutine, which allows us to exit gracefully, killing
# all abandoned jobs and wiping all semaphores.
```

```
#
# Signal 15 is given by PBS when time expires, and is the signal
# given by default when you use qsig (without specifying a signal).
#####
trap 'echo "Received signal 15"; Cleanup 1' 15



###################### Begin job header -- DO NOT MODIFY
###################

$ECHO ""
$ECHO
"#############################################################
##"
$ECHO "Job started at `date`"
$ECHO
"#############################################################
##"
$ECHO ""
$ECHO
"#############################################################
##"
$ECHO "Working directory is $PBS_O_WORKDIR"
$ECHO ""
$ECHO "Job started on host `hostname`"

EXEC_NODE=`hostname -s`

$ECHO "The hostname is"
$ECHO `hostname`
$ECHO ""
$ECHO "Job running on node(s):"
$CAT $PBS_NODEFILE
$ECHO
"#############################################################
##"
$ECHO ""

################################### End job header
############################


### Necessary variable for mpi to run correctly -- do not delete
### This is from ASPEN
P4_GLOBMEMSIZE=32000000
export P4_GLOBMEMSIZE

## Required to run g98 on aspen
eval `$MODULE add mpich-eth-intel`
eval `$MODULE add pgi`
```

```
### Define variable for local storage on compute nodes associated with the job
TEMPDIR="/home/sv4e/Trial_Gaussian"


## Change to the working directory
cd $TEMPDIR

### Now copy the input files
mkdir scr1 scr2 scr3 scr4

for theFile in $BASENAME ; do
   $ECHO "Starting molecule $theFile"

   # $CP $PBS_O_WORKDIR/${theFile}.gjf $TEMPDIR/${theFile}.gjf

   ## This Perl statement does a DOS-to-Un*x conversion of the input file.
   ## See: http://www.technocage.com/~caskey/dos2unix/ for details.
   $PERL -pi -e 's/\r\n/\n/;' $TEMPDIR/${theFile}.gjf
   $ECHO "" >> $TEMPDIR/${theFile}.gjf
   $G98 ${theFile}.gjf ${theFile}.out
done



Cleanup

#####
# NOTE: Cleanup exits the script, so nothing below the Cleanup line will
# *ever* get run. Don't bother to put anything down here.
#####
```